UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte Paul Reuben Day, Brian Robert Muras, Anne Marie Ryg

Appeal No. _____
Application No. 10/754,010

APPEAL BRIEF

Attorney Docket No. ROC920030366US1 PATENT
Confirmation No. 7130

CERTIFICATE OF ELECTRONIC TRANSMISSION
I hereby certify that this correspondence for Application No. 10/754,010 is being electronically transmitted to Technology Center 2167 via EFS-WEB, on December 9, 2008.

/Scott A. Stinebruner/ December 9, 2008
Scott A. Stinebruner, Reg. No. 38,323 Date

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Paul Reuben Day et al.	Art Unit:	2167
Application No.:	10/754,010	Examiner:	Kimberly M. Lovel
Filed:	January 8, 2004		
For:	METHOD AND SYSTEM FOR A SELF-HEALING QUERY ACCESS PLAN		

Mail Stop Appeal Brief - Patents
Commissioner for Patent
P.O. Box 1450
Alexandria, VA 22213-1450

APPEAL BRIEF

I. REAL PARTY IN INTEREST

This application is assigned to International Business Machines Corporation, of Armonk, New York.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1, 3-8 and 10-12 are pending in the Application, stand rejected, and are now on appeal. Claims 2, 9 and 13-21 have been canceled without prejudice.

IV. STATUS OF AMENDMENTS

There have been no amendments filed subsequent to the final rejection mailed October 6, 2008.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Applicant's invention is generally directed to a database engine and optimizer framework that support the ability to automatically respond to execution errors that occur during execution of a query plan to allow for continued execution of a query plan. (Application, page 6, lines 2-4).

Specifically, conventional database management systems rely on query optimizers to choose the most efficient way to execute a database query. The output of a query optimizer is typically referred to as an "execution plan," "access plan," or just "plan," which typically incorporates low-level information telling the database engine that ultimately handles a query precisely what steps to take (and in what order) to execute the query. (Application, page 2, lines 20-24). Due to the fact that number of possibly query forms that can be used to represent a given query, and due to the need to respond relatively quickly to queries, a query optimizer often has only a limited amount of time to pare the search space of possible execution plans down to an optimal plan for a particular query. (Application, page 3, lines 3-16). As a result, in many instances a query optimizer will not generate the perfect plan by which to execute a query.

In some instances, a query plan may encounter errors during execution, some of which may be so significant as to require termination of execution of the query. These errors, which are termed execution errors, and which may also be referred to as function checks, cause the execution of a query to be halted when they are encountered. Conventional database engines include built-in error reporting and error handling facilities, and will typically report an error that includes some type of identifying code as well as the location in the query where the error occurred. (Application, page 3, lines 17-21). However, these engines do not have any ability to repair or correct the error without manual intervention. As a result, in response to such an error, the next step for the database user typically involves calling a customer support engineer and

trying to resolve the problem via telephone or e-mail, leading to user frustration, system downtime, and lower productivity.

In contrast, embodiments consistent with the invention attempt to minimize the impact of execution errors that occur during the processing of the query plan for a database query through the use of a self-healing database engine and optimizer framework that support automatically responding to execution errors to allow continued execution of a query plan. (Application, page 4, lines 2-4). Upon encountering an execution error of the type that ordinarily halts execution of a query plan, the database engine automatically initiates a rebuilding of the query plan and executes the rebuilt query plan. (Application, page 4, lines 4-6). In some embodiments, if an error is encountered in the rebuilt query plan then the query implementation methods are analyzed, and if a query function is identified for which an alternative implementation method is available, then this alternative implementation method is substituted to create a new query plan. The new query plan is then executed to determine if the error is corrected. (Application, page 4, lines 6-10).

For the convenience of the Board, independent claims 1, 7 and 12 have been reproduced below and annotated with references to the specification and drawings to satisfy the requirement to concisely explain the claimed subject matter:

Independent Claim 1

> A method (Application, Fig. 3, page 9, line 27 to page 12, line 9) for automatic handling of errors within a database engine (Application, Fig. 2, block 44, page 8, lines 14-15), the method comprising the steps of:
>
> detecting an error while executing a query access plan (Application, Fig. 2, block 50, Fig. 3, block 304, page 8, lines 12-14, page 10, lines 4-6), wherein the error is an execution error of a type that halts execution of the query access plan (Application, page 10, lines 4-6), and wherein the query access plan is of the type generated by a query optimizer (Application, Fig. 2, blocks 42, 50, page 8, lines 10-14);

in response to detecting the error, automatically rebuilding the query access plan with the query optimizer to generate a new query access plan (Application, Fig. 3, block 308, page 10, lines 13-23); and

executing the new query access plan to generate at least a portion of a result set for storage or display (Application, Fig. 3, blocks 310, 312, page 10, lines 24-27).

Independent Claim 7

A method (Application, Fig. 3, page 9, line 27 to page 12, line 9) for automatic handling of errors within a database engine (Application, Fig. 2, block 44, page 8, lines 14-15), the method comprising the steps of:

receiving an error while executing a function within a query access plan (Application, Fig. 2, block 50, Fig. 3, block 304, page 8, lines 12-14, page 10, lines 4-6, page 11, lines 18-20), wherein the error is an execution error of a type that halts execution of the query access plan (Application, page 10, lines 4-6), and wherein the query access plan is of the type generated by a query optimizer (Application, Fig. 2, blocks 42, 50, page 8, lines 10-14);

identifying a first implementation method of the function within the query access plan (Application, Fig. 3, blocks 316-318, page 11, lines 18-23);

rebuilding the query access plan with the query optimizer by replacing the first implementation method with a second implementation method of the function so as to generate a new query access plan (Application, Fig. 3, block 318, page 11, lines 20-25); and

executing the new query access plan to generate at least a portion of a result set for storage or display (Application, Fig. 3, blocks 310, 312, page 10, lines 24-27, page 12, lines 3-7).

Independent Claim 12

A method (Application, Fig. 3, page 9, line 27 to page 12, line 9) for automatic handling of errors within a database engine (Application, Fig. 2, block 44, page 8, lines 14-15), the method comprising the steps of:

executing a query access plan (Application, Fig. 2, block 50, Fig. 3, block 302, page 10, lines 1-2) comprising a plurality of functions (Application, page 11, lines 6-17, page 12, lines 10-26), each function including a first implementation method (Application, page 11, lines 11-23), and the query access plan of the type generated by a query optimizer (Application, Fig. 2, blocks 42, 50, page 8, lines 10-14);

detecting a first error when executing a first function (Application, Fig. 3, block 304, page 10, lines 4-6), wherein the first error is an execution error of a type that halts execution of the query access plan (Application, page 10, lines 4-6);

rebuilding the query access plan with the query optimizer to generate a new query access plan (Application, Fig. 3, block 308, page 10, lines 13-23);

executing the new query access plan to generate at least a portion of a result set for storage or display (Application, Fig. 3, blocks 310, 312, page 10, lines 24-27);

receiving a second error while executing the first function within the new query access plan (Application, Fig. 3, block 316, page 11, lines 18-20); and

rebuilding the new query access plan by replacing the first implementation method with a second implementation method of the function (Application, Fig. 3, block 318, page 11, lines 20-25).

Other support for the claimed subject matter may generally be found in Fig. 3 and the accompanying text at pages 9-14 of the Application as filed. In addition, it should be noted that, as none of the claims recite any means plus function or step plus function elements, no

identification of such elements is required pursuant to 37 CFR §41.37(c)(1)(v). Furthermore, there is no requirement in 37 CFR §41.37(c)(1)(v) to provide support for the subject matter in the separately argued dependent claims, as none of these claims recite means plus function or step plus function elements, and so no discussion of any of these claims is provided.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claims 1, 3-8 and 10-12 are rejected to under 35 U.S.C. § 103 (a) as being unpatentable over the article "Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans" by Kabra et al. (hereinafter "Kabra") in view of Ziauddin et al. (USPG Pub. No. 2005/0177557) (hereinafter "Ziauddin").

VII. ARGUMENT

Applicant respectfully submits that the Examiner's rejections of claims 1, 3-8 and 10-12 are not supported on the record, and should be reversed.

A. Claims 1, 3-8 and 10-12 are not unpatentable over Kabra in view of Ziauddin

Claims 1, 3-8 and 10-12 stand rejected as being unpatentable over Kabra in view of Ziauddin. Applicant respectfully submits, however, that in the instant case, the Examiner has failed to establish a *prima facie* case of obviousness as to claims 1, 3-8 and 10-12, and thus, the rejections thereof should be reversed.

Based on the Supreme Court's decision in KSR, 127 S. Ct. at 1734, a *prima facie* showing of obviousness still requires that the Examiner establish that the differences between a claimed invention and the prior art "are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art." 35 U.S.C. §103(a). Such a showing requires that all claimed features be disclosed or suggested by the prior art. Four factors generally control an obviousness inquiry: 1) the scope and content of the prior art; 2) the differences between the prior art and the claims; 3) the level of ordinary skill in the pertinent art; and 4) secondary considerations of non-obviousness, such as commercial success of products covered by the patent claims, a long felt but unresolved need for the invention, and failed attempts by others to make the invention. KSR, 127 S. Ct. at 1734 (quoting Graham v. John Deere Company, 383 U.S. 1, 17-18 (1966)) ("While the sequence of these

questions might be reordered in any particular case, the [Graham] factors continue to define the inquiry that controls.").

Moreover, in KSR, the Court explained that "[o]ften, it will be necessary for a court to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in the art, all in order to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue" and "[t]o facilitate review, this analysis should be made explicit." KSR, 127 S. Ct. at 1740-41 *quoting* In re Kahn, 441 F.3d 977, 988, 78 USPQ2d 1329, 1336 (Fed. Cir. 2006) ("[R]ejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness."). But, not every combination is obvious "because inventions in most, if not all, instances rely upon building blocks long since uncovered, and claimed discoveries almost of necessity will be combinations of what, in some sense, is already known." KSR, 127 S. Ct. at 1741.

As a result, after KSR, while there is no rigid requirement for an explicit "teaching, suggestion or motivation" to combine references, there still must be some evidence of "a reason that would have prompted a person of ordinary skill in the relevant field to combine the elements in the way the claimed new invention does" in an obviousness determination. KSR, 127 S. Ct. at 1731.

Applicant respectfully submits that, in the instant case, the Examiner has failed to establish a *prima facie* case of obviousness as to claims 1, 3-8 and 10-12, and as such, the rejections thereof should be reversed. Applicant's remarks in rebuttal to the Examiner's rejections are presented below, starting with the relevant independent claims and followed by a discussion of selected dependent claims. In some cases, specific discussions of particular claims are not made in the interest of streamlining the appeal. The omission of a discussion with respect to any particular claim, however, should not be interpreted as an acquiescence as to the merits of the Examiner's rejection of the claim, particularly with respect to claims reciting features that are addressed in connection with the rejection applied to other claims pending in the appeal.

Independent Claim 1

Claim 1 recites a method for automatic handling of errors within a database engine, which includes detecting an error while executing a query access plan; in response to detecting the error, automatically rebuilding the query access plan with the query optimizer to generate a new query access plan; and executing the new query access plan to generate at least a portion of a result set for storage or display. The claim also recites that the error is an execution error of a type that halts execution of the query access plan, and that the query access plan is of the type generated by a query optimizer.

Notably, the error that is addressed in claim 1 is specifically an execution error of a type that halts execution of the query access plan. As described in page 10, lines 1-6 of the Application, during execution, the database engine 44 may, in step 304, detect an error that halts execution of the query. Within some database environments, such an error is known as a function check. A number of different such errors may be detected; some relate to opening various tables; some relate to formatting the result set; some relate to problems with selection criteria; and many other types of errors are possible as well. Moreover, as indicated on page 12, lines 21-23 of the Application, execution errors sometimes occur because of problems in the database engine 44 or the optimizer 42, with a "patch" typically issued from the database vendor that addresses the problems. Nonetheless, the claimed execution error halts execution of a query, and is generally a type of involuntary operation that essentially halts a query and halts its continued execution.

In rejecting claim 1, the Examiner relies on Kabra and Ziauddin. The Examiner asserts that Kabra discloses limitations of the claim 1 at the abstract, lines 6-8 and page 109, col. 2, line 34 to page 110, col. 1, line 15. The Examiner admits, however, that Kabra fails to explicitly disclose the limitation "wherein the error is an execution error of a type that halts execution of the query access plan," required by claim 1.

For this, the Examiner cites Ziauddin, and specifically the abstract and paragraphs [0003] and [0017]. The Examiner also asserts that Ziauddin also discloses "in response to detecting the error, automatically rebuilding the query access plan to generate a new query plan," citing in particular paragraph [0017].

However, contrary to the Examiner's assertion, Ziauddin adds nothing to the rejection because, just like Kabra, Ziauddin does not disclose or suggest an execution error that halts execution of a query access plan. The errors that are addressed in Ziauddin are runaway errors, which result when a query runs longer than expected (paragraph [0017]).

Indeed, if anything the runaway errors of Ziauddin are the precise opposite of an "execution error that halts execution of a query access plan." As indicated in paragraph [0004] of Ziauddin, a "...suboptimal plan results in a run-away execution of the query...[in] other words, the plan, when executed, causes a SQL statement to run for a long time with enormous use of system resources." Therefore, runaway errors, as opposed to halting execution, cause execute to continue an inordinately long time. There is nothing associated with a runaway error that causes the execution of a query access plan to be halted. In fact, a runaway error prevents a query access plan from stopping or completing.

In addition, the disclosure of Ziauddin is more in line with optimizing sub-optimal query plans, like Kabra, because Ziauddin monitors these runaway errors, and only the longest running queries may be selected for reoptimization, leaving the others to continue to execute. Indeed, paragraphs [0003]-[0004] of Ziauddin disclose the problem of suboptimal plans, and the Abstract and paragraph [0017] go on to disclose an automatic tuning optimizer (ATO) that, in a background process, optimizes the execution plan for queries having runaway execution. Ziauddin discloses that query executions having a negative execution time difference can be automatically identified as a runaway query execution. Next, the ATO performs various analyses of the corresponding SQL statement such as automatic identification and correction of incorrect statistics, cardinality estimates, and cost estimates related to the statement. Ziauddin also discloses that the ATO builds a new plan, and compares the estimated execution time of the new plan to the remaining execution time of the current plan, and if the estimated time for the new plan is less than the remaining time for the current plan, the current plan is aborted and replaced with the new plan.

Thus, while the Examiner does correctly point out that the Abstract of Ziauddin teaches execution of query plans and paragraph [0017] teaches aborting a current plan and replacing it with a new plan, both of these features are in the context of runaway errors that continue the

execution of a query, and NOT execution errors that halt execution of a query and the associated access query plan, as required by claim 1.

It may be that the Examiner is taking the position that since Ziauddin discloses aborting a query due to a runaway error, this somehow discloses an execution error that halts the execution of a query. However, claim 1 requires that the "error" halt the execution of a query. In Ziauddin, the optimizer aborts a query after selecting an appropriate candidate, so it is the optimizer that actually halts the query. Claim 1 requires that the error be "an execution error of a type that halts execution of the query access plan," and any error that did not halt execution of a query access plan in all circumstances would not meet this limitation of claim 1.

Indeed, the aborting and substitution for the new plan in Ziauddin is more or less voluntary, based on statistics and estimated times, because execution of a runaway query can and will often result in the current plan continuing and completing, as indicated in block 160 and paragraphs [0015]-[0017]. Block 160 of Fig. 1 indicates that the process "determine[s] whether to abort the current plan and execute the new plan." Moreover, paragraphs [0015] and [0017] indicate that the current plan MAY be aborted but does not need to be. For instance, paragraph [0015] states that "[t]he automatic prevention of run-away query executions may abort a current execution of a query run if the automatic process has produced an improved plan in the background, and further, has determined a benefit to aborting the current execution and performing an execution of the new plan." Thus, aborting the current plan is not automatic.

It also is important to note that the early exit by aborting stems from runaway errors causing the extended execution of runaway queries due to nonvalidated statistics, and NOT from the actual execution of a query. Furthermore, any error in statistics collection leading to the runaway errors does not cause a query access plan to be automatically rebuilt, as is required by claim 1.

Moreover, even if an access plan is aborted in Ziauddin, it is because the runaway error causing the query to continue to execute for too long and using too many resources (paragraph [0004]), not because of an execution error that causes the query (and query access plan) to stop executing. Similarly, the detection of a sub-optimal query, as occurs in Kabra, perhaps results in

a query being modified or re-optimized, but the decision to perform such a modification or optimization is made by the optimizer, and is more or less a voluntary operation performed on the basis of collected performance statistics. Execution errors that halt execution of a query, on the other hand, are involuntary operations that essentially terminate a query and prohibit its continued execution. And, Applicant can find no teaching in either reference purporting to disclose or suggest specifically performing an automated rebuild of a query access plan "in response to" a detected error that halts execution of the query access plan.

The runaway errors, in fact, do not even appear to be execution-related errors, since even when such "errors" exist, the queries apparently complete at some point in the future. Were the runaway errors in Ziauddin of the type that caused a query execution engine to "hang" and become unresponsive, these errors might arguably construed as "execution" errors. However, given the runaway errors are only associated with sub-optimal query execution, they are not properly considered to be execution errors of the type contemplated by Applicant's claims.

The Examiner does attempt to rebut Applicant's arguments with respect to claim 1 on pages 8-10 of the Final Office Action dated October 6, 2008. The Examiner argues that the broadest reasonable interpretation of the term "error" is broad enough to include "a query that has been executing longer than predicted." However, the Examiner's interpretation ignores the qualifier in claim 1 that the error be "an execution error of a type that halts execution of the query access plan." As discussed above, a runaway error is not an "execution error," as it is related more to a sub-optimal query access plan. Moreover, Applicant submits that a runaway error, which under certain circumstances will cause the ATO to abort a query, is not of the type that "halts execution of [a] query access plan." The fact that the ATO later makes as decision as to whether to halt a query breaks the chain of causation from the runaway error to the halt in execution.

Applicant also notes that the rebuilding of the query plan in claim 1 is performed specifically "in response to detecting the error." The Examiner's interpretation of "error," however, effectively reads this limitation out of claim 1 because the error is being conflated with the rebuild process that is performed in Ziauddin when determining whether to abort a query.

In addition, since the error "halts execution of the query access plan," it is inherent that the detection of the error in claim 1 occurs in connection with the execution of the query access plan having been halted due to the error. Furthermore, since the automatic rebuild of the query access plan is performed specifically "in response to detecting the error," the rebuild of the query access plan occurs after the execution of the query access plan has been halted due to the error. Ziauddin, in contrast, only halts execution of a query access plan after the query access plan has been rebuilt and a cost comparison has been made, as illustrated in Fig. 1, blocks 150, 160 and discussed in paragraph [0021].

The Examiner also argues on page 9 that while the optimizer in Ziauddin does halt execution of a plan, the error does still halt execution of the query. The Examiner also asserts that the claim fails to limit the means of how the error halts execution. Applicant respectfully submits, however, that since claim 1 does not make the error conditional upon anything else in order to halt execution, and that the rebuild is performed "in response to" the error, the unconditional nature of the error, and the automatic response generated thereby, distinguishes from the runaway errors of Ziauddin. Applicant also notes that claim 3 does in fact recite a specific type of error (a function check) which one of ordinary skill in the art would recognize as being a type of error that unconditionally halts execution of a query, yet the Examiner has rejected that claim based upon language in Kabra that runs completely contrary to the Examiner's admission that Kabra does not disclose an error that halts execution of a query plan. Applicant would be open to amending claim 1 to further clarify the unconditional nature of the recited error should the Examiner wish to propose such language; however, Applicant submits that the claim language as it now stands is sufficient to distinguish from the types of errors disclosed in Kabra and Ziauddin. For example, Applicant would consider an amendment that the error be of the type that "always" halts execution of the query plan would be redundant, since definiteness in a claim usually demands that the claimed limitations not be optional.

Therefore, while Kabra and Ziauddin both generally attempt to improve optimization of database queries, neither reference, alone or in combination, discloses or suggests doing so for "an execution error of a type that halts execution of the query access plan", among other limitations in claim 1. Applicant accordingly submits that the combination proposed by the

Examiner does not disclose or suggest each and every limitation in claim 1, and therefore, Applicant respectfully submits that independent claim 1 is novel and non-obvious over Kabra and Ziauddin. Reversal of the Examiner's rejection of claim 1, and allowance of that claim as well as of claims 3-6 which depend therefrom, are therefore respectfully requested.

Independent Claim 7

Claim 7 generally recites a method for automatic handling of errors within a database engine. This method includes receiving an error while executing a function within a query access plan. The error is an execution error of a type that halts execution of the query access plan, and the query access plan is of the type generated by a query optimizer. The method also includes identifying a first implementation method of the function within the query access plan, rebuilding the query access plan with the query optimizer by replacing the first implementation method with a second implementation method of the function so as to generate a new query access plan, and executing the new query access plan to generate at least a portion of a result set for storage or display.

In rejecting claim 7, the Examiner also relies on the same arguments and citations of Kabra and Ziauddin made in connection with claim 1. The Examiner admits once again, however, that Kabra fails to explicitly disclose the limitation "wherein the error is an execution error of a type that halts execution of the query access plan". The Examiner also admits that Kabra does not disclose "identifying a first implantation method of the function within the new query access plan" and "rebuilding the new query access plan by replacing the first implementation method with a second implementation method of the function so as to generate a rebuilt query access plan". For the latter two limitations, the Examiner cites Ziauddin, and in particular, paragraphs [0029] and [0017].

As argued above, the combination of Kabra and Ziauddin does not disclose an execution error of the type recited in Applicant's claims, and additionally, does not disclose functions and replacing the first implementation method with a second implementation method of the function.

First, paragraphs [0029] and [0017] of Ziauddin merely disclose statistics and profiling for generating the new access plan in the background by the ATO. However, these paragraphs

do NOT teach or suggest any functions or the function related implementations. Statistical and profiling information are helpful and often used to select an implementation method, and may even be stored in an implementation method (e.g., in an index), but statistical and profiling information are NOT implementation methods. As described in page 12, lines 8-24 of the Application:

> In a robust database system there are hundreds of different query functions with alternative implementation methods that an optimizer chooses from when building a query plan. Even though an exhaustive list of the different functions and their alternative implementation methods are not explicitly included herein, the above-described method contemplates, when appropriate, rebuilding a query plan with alternative implementation methods for any of the wide variety of functions likely to be encountered in a query plan. For example, an ordering function may be implemented using "order by index" or "order by sort"; a grouping function may be implemented using "group by hash" or "group by index"; and a join function may be implemented using either a nested loop method or a hashing method. Additionally, when handling a subquery function, the optimizer may implement it as a "join" or as a separate subquery; or when coding a view function in the query plan, the optimizer can implement the view function using a method known as "materialization" or an alternative method known as "merge". Other exemplary functions include such functions as hash, view, host variables, parameter markers, literals, nested loop, materialization, sort, bit map, sub query, sparse index, merge, and table scan. These are just a few exemplary functions for which an optimizer selects from among alternative implementation methods when generating a query plan.

Second, there is no disclosure in Ziauddin of replacing implementation methods related to functions. For example, cited paragraph [0017] explicitly indicates that "if the execution plan built by the ATO is different from the one that is currently executing, the ATO can estimate…", which implies that the new access plan created by the ATO is NOT always different from the current plan. And if the new access plan generated by the ATO is not different, then the same implementation methods are being used, therefore, there is no replacement of a first implementation method with a second implementation method. In instances that the new access plan is different from the current access plans, the implementation methods can be different but there is no disclosure in Ziauddin affirmatively teaching or suggesting this, let alone with respects to functions.

In the Final Office Action, and specifically on page 10, the Examiner argues that the collection of statistics in Ziauddin is correlated to the execution of a function. The Examiner also states that "[h]ow the statistics are implemented the first time is considered to represent the first implementation and when they are corrected, it is considered to be the second implementation." With all due respect, the Examiner's tortured attempt to analogize the collection of statistics to a function in a query plan completely distorts the concept of a function beyond that which would be recognizable by one of ordinary skill in the art. Applicant has set forth the concept of a function with a great deal of clarity in the specification, and it should be evident from a reading of the passage quoted above that the collection of statistics is by no means analogous to any of the listed types of functions, all of which are types of operations that are part and parcel of a query access plan. Under any reading that is both reasonable and within the context of the Application, the collection of statistics would not be considered to be a "function" by one of ordinary skill in the art.

Furthermore, the Examiner refers to "implementing" statistics differently at different times. Applicant is not aware of any different "implementation" of a statistic. Statistics may vary from time to time based upon changes to the underlying data in a database; however, those differences are not differences in "implementation" since they are collected in the same way at those different times. The American Heritage dictionary defines implement as "[t]o put into practical effect; carry out: implement the new procedures." Applicant submits that collecting statistics at different times, which vary only due to changes in the underlying data, does not correspond to different implementations of a function. In fact, the collection of statistics at different times in all likelihood utilizes the same methodology (i.e., implementation). Ziauddin does not even disclose that at different times, different methodologies can be used to collect statistics, which would seem to be a threshold requirement for arguing a correspondence between the collection of statistics and the implementation of a function.

Indeed, while Kabra and Ziauddin both generally attempt to improve optimization of database queries, neither reference, alone or in combination, discloses or suggests "wherein the error is an execution error of a type that halts execution of the query access plan"; "identifying a first implantation method of the function within the new query access plan"; and "rebuilding the

new query access plan by replacing the first implementation method with a second implementation method of the function so as to generate a rebuilt query access plan". Applicant accordingly submits that the combination proposed by the Examiner does not disclose or suggest each and every limitation in claim 7, and therefore, Applicant respectfully submits that independent claim 7 is novel and non-obvious over Kabra and Ziauddin. Reversal of the Examiner's rejection of claim 7, and allowance of the claim and of claims 8 and 10-11 which depend therefrom, are therefore respectfully requested.

Independent Claim 12

Claim 12 generally recites a method for automatic handling of errors within a database engine. The method includes executing a query access plan comprising a plurality of functions, each function including a first implementation method, and the query access plan of the type generated by a query optimizer, detecting a first error when executing a first function. The first error is an execution error of a type that halts execution of the query access plan. The method also includes rebuilding the query access plan with the query optimizer to generate a new query access plan, executing the new query access plan to generate at least a portion of a result set for storage or display, receiving a second error while executing the first function within the new query access plan, and rebuilding the new query access plan by replacing the first implementation method with a second implementation method of the function.

In rejecting claim 12, the Examiner also relies on the same arguments and citations of Kabra and Ziauddin made in connection with claims 1 and 7. The Examiner once again admits, however, that Kabra fails to explicitly disclose the limitation "wherein the error is an execution error of a type that halts execution of the query access plan". The Examiner also admits that Kabra does not disclose "receiving a second error while executing the first function within the new query access plan" and "rebuilding the new query access plan by replacing the first implementation method with a second implementation method of the function". For these latter two limitations, the Examiner cites Ziauddin, and in particular, paragraphs [0029] and [0017].

Claim 12 is at least patentable over the prior art for the reasons presented in connection with claims 1 and 7 above. In addition, as argued above in connection with claim 7, paragraphs

[0029] and [0017] of Ziauddin do not disclose or suggest any functions or function related implementations, let alone disclosing or suggesting a plurality of functions, a second function, and additional function related implementations for the second function. This claim generally recites in part the detection of two errors, a first error that results in the rebuilding of a query access plan, and then a second error occurring during the execution of the rebuilt query access plan, and the handling of the latter error by rebuilding the query access plan to replace a first implementation method of a function with a second implementation method of the function. Applicant can find no disclosure or suggestion in either Ziauddin or Kabra directed to any type of multi-stage error processing method that handles subsequent errors that occur after an attempt has already been made to rectify an error in a query access plan, much less any method that handles a subsequent error in the specific manner recited in claim 12.

Even assuming *arguendo* that the handling of errors in Ziauddin and Kabra is analogous to automatically rebuilding a query access plan in response to a detected error, as asserted by the Examiner, neither reference addresses any functionality for handling a subsequent error that occurs after a query access plan has been rebuilt and re-executed. Accordingly, Applicant submits that claim 12 is additionally patentable over the cited references for this additional reason.

Indeed, while Kabra and Ziauddin both generally attempt to improve optimization of database queries, neither reference, alone or in combination, discloses or suggests "wherein the error is an execution error of a type that halts execution of the query access plan"; "identifying a first implantation method of the function within the new query access plan"; "rebuilding the new query access plan by replacing the first implementation method with a second implementation method of the function so as to generate a rebuilt query access plan"; "receiving a second error while executing the first function within the new query access plan"; and "rebuilding the new query access plan by replacing the first implementation method with a second implementation method of the function" Indeed, independent claim 12 has additional function related limitations that the Examiner cannot ignore, and even in the Final Office Action, the Examiner fails to even attempt to address this additional language, relying solely on the arguments made in connection with claims 1 and 7.

Applicant accordingly submits that the combination proposed by the Examiner does not disclose or suggest each and every limitation in claim 12, and therefore, Applicant respectfully submits that independent claim 12 is novel and non-obvious over Kabra and Ziauddin. Reversal of the Examiner's rejection of claim 12, and allowance of the claim, are accordingly requested.

Dependent Claim 3

Claim 3 depends from claim 1 and additionally recites that the error is a function check. The Examiner cites Kabra at page 109, column 2, lines 29-33 for allegedly disclosing this feature. However, the cited disclosure at page 109 does not disclose a function check, which is recognized in the art, and specifically defined in the specification, as a type of error that halts execution in a database engine. Moreover, the Examiner's assertion that Kabra discloses a function check is completely inconsistent with the Examiner's admission (made in connection with the rejections of claims 1, 7 and 12) that Kabra does not disclose an error that halts execution of a query plan.

In addition, as argued above in connection with claims 7 and 12, Ziauddin also does not teach or suggest functions or anything related to functions, nor does Ziauddin disclose any error that is properly analogized to a function check. Furthermore, even if the Examiner attempts to argue that the runaway errors of Ziauddin somehow halt execution of a query because an optimizer may voluntarily select a runaway query to be aborted, these errors are not properly classified as function check errors (and indeed, are completely opposite thereto because they continue to run longer than expected). Applicant respectfully submits that when claim 3 is read in context with the limitations of claim 1 from which it depends, the Examiner's rejection cannot be sustained. Reversal of the Examiner's rejection of claim 3, and allowance of the claim, are therefore respectfully requested.

Dependent Claims 4 and 10

Claims 4 and 10, which depend from claims 1 and 7, respectively, recite to varying extents the concept of handling another error detected while executing a query access plan that has been automatically rebuilt in response to an execution error by rebuilding the query access plan to replace a first implementation method of a function with a second implementation

method. Claim 4 is representative, and recites receiving another error while executing a function within the new query access plan, identifying a first implementation method of the function within the new query access plan, and rebuilding the new query access plan by replacing the first implementation method with a second implementation method of the function so as to generate a rebuilt query access plan. Claim 10 adds the concept of replacing a second implementation of a function with a third implementation in response to an additional error.

In rejecting claim 4, the Examiner relies on paragraph [0029] of Ziauddin, relating to the creation of the new access plan by the ATO. The new access plan is created from the statistics, estimates, and settings analyses to be stored in a SQL Profile. Once the SQL Profile is created, it is used in conjunction with the existing statistics by the compiler to produce a well-tuned plan for the corresponding SQL statement. As argued above in connection with claim 7, paragraph [0017] of Ziauddin implies that the new access plan may be the SAME as the current plan, with the same implementation methods, therefore, there is teaching or suggestion of replacing the implementation method. Moreover, Ziauddin does not disclose the function related limitations.

In addition, as noted above in connection with claim 12, irrespective of whether any of the operations disclosed in Ziauddin can be analogized to functions or function implementations, neither reference discloses a multi-stage error handling protocol capable of handling multiple errors that are encountered during the execution of a query plan. Reversal of the Examiner's rejections of claims 4 and 10, and allowance of those claims, are therefore respectfully requested.

Dependent Claims 5-6 and 11

Claims 5 and 6, and claim 11, which depend from claims 1 and 7, respectively, recite to varying extents logging information about at least one error and the new query access plan. Claim 11 is representative, and recites logging information about the error, the another error, and the new query access plan.

In rejecting claim 11, Examiner relies on page 109, column 1, lines 16-27 of Kabra, relating to gathering statistics about intermediate query results during the course of query execution to be used to improve the estimates of the query optimizer. These improved estimates can be used to improve the allocation of memory to the various operators of the query.

Kabra indicates that when improved estimates are available, the memory management module can be re-invoked and supplied with the new estimates. The memory management module used these new estimates to produce a new memory allocation for the remainder of the query. Overall performance is expected to improve since the new memory allocation is based on improved estimates.

However, none of this discloses logging information about errors and the new query access plan. Logging estimates or statistics is NOT the same as logging information about errors and the query access plan. The cited references therefore fail to disclose or suggest the additional features recited in claims 5-6 and 11. Reversal of the Examiner's rejections of these claims, and allowance of these claims, are therefore respectfully requested.

Dependent Claim 8

Claim 8 is not argued separately.

B. Conclusion

Applicant respectfully requests that the Board reverse the Examiner's rejections of claims 1, 3-8, and 10-12, and that the Application be passed to issue. If there are any questions regarding the foregoing, please contact the undersigned at 513/241-2324. If any other charges or credits are necessary to complete this communication, please apply them to Deposit Account 23-3000.

Respectfully submitted,

December 9, 2008
Date

/Scott A. Stinebruner/
Scott A. Stinebruner
Reg. No. 38,323
WOOD, HERRON & EVANS, L.L.P.
2700 Carew Tower
441 Vine Street
Cincinnati, Ohio 45202
Telephone: (513) 241-2324
Facsimile: (513) 241-6234

VIII. CLAIMS APPENDIX: CLAIMS ON APPEAL (S/N 10/754,010)

Listing of Claims:

1. (Previously Presented) A method for automatic handling of errors within a database engine, the method comprising the steps of:

detecting an error while executing a query access plan, wherein the error is an execution error of a type that halts execution of the query access plan, and wherein the query access plan is of the type generated by a query optimizer;

in response to detecting the error, automatically rebuilding the query access plan with the query optimizer to generate a new query access plan; and

executing the new query access plan to generate at least a portion of a result set for storage or display.

2. (Cancelled)

3. (Original) The method of claim 1, wherein the error is a function check.

4. (Original) The method of claim 1 further comprising the steps of:

receiving another error while executing a function within the new query access plan;

identifying a first implementation method of the function within the new query access plan; and

rebuilding the new query access plan by replacing the first implementation method with a second implementation method of the function so as to generate a rebuilt query access plan.

5. (Original) The method according to claim 1, further comprising the step of:

logging information about the error, and the new query access plan.

6. (Original) The method according to claim 1, further comprising the step of:

reporting the error.

7. (Previously Presented) A method for automatic handling of errors within a database engine, the method comprising the steps of:

receiving an error while executing a function within a query access plan, wherein the error is an execution error of a type that halts execution of the query access plan, and wherein the query access plan is of the type generated by a query optimizer;

identifying a first implementation method of the function within the query access plan;

rebuilding the query access plan with the query optimizer by replacing the first implementation method with a second implementation method of the function so as to generate a new query access plan; and

executing the new query access plan to generate at least a portion of a result set for storage or display.

8. (Original) The method of claim 7, wherein the function is one of a join function, an indexing function, a grouping function, and an ordering function.

9. (Cancelled)

10. (Previously Presented) The method of claim 7, further comprising the steps of:

receiving another error while executing the function within the new query access plan; and

rebuilding the new query access plan by replacing the second implementation method with a third implementation method of the function.

11. (Original) The method according to claim 10 further comprising the step of:

logging information about the error, the another error, and the new query access plan.

12. (Previously Presented) A method for automatic handling of errors within a database engine, the method comprising the steps of:

executing a query access plan comprising a plurality of functions, each function including a first implementation method, and the query access plan of the type generated by a query optimizer;

detecting a first error when executing a first function, wherein the first error is an execution error of a type that halts execution of the query access plan;

rebuilding the query access plan with the query optimizer to generate a new query access plan;

executing the new query access plan to generate at least a portion of a result set for storage or display;

receiving a second error while executing the first function within the new query access plan; and

rebuilding the new query access plan by replacing the first implementation method with a second implementation method of the function.

13.- 21 (Cancelled)

IX. EVIDENCE APPENDIX

[APPLICATION NO. 10/754,010]

None.

X. RELATED PROCEEDINGS APPENDIX

[APPLICATION NO. 10/754,010]

None.